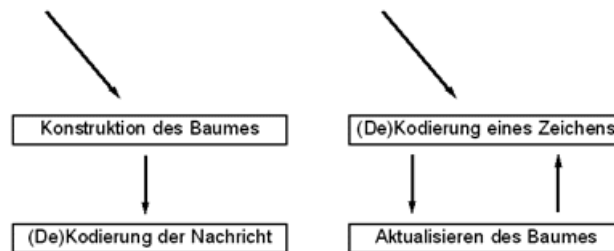


Freiwillige Übung zum Thema: Adaptive Huffman-Kodierung

Hintergrund-Info: Adaptive Verfahren (Quelle: Uni Karlsruhe)

Die einfache Huffman-Kodierung verwendete für die Konstruktion der Codes die Häufigkeiten aus dem gesamten Text. Das machte es notwendig, diese Häufigkeiten mit der kodierten Nachricht zu übermitteln.

Die Idee eines sogenannten adaptiven Verfahrens ist, zu jedem Zeitpunkt nur den Baum zu verwenden, der sich aus den bis dahin kodierten bzw. dekodierten Zeichen ergibt. Das bedeutet, daß nach der Kodierung bzw. nach der Dekodierung jedes Zeichens die Häufigkeiten aktualisiert werden und der zugehörige Baum überprüft werden müssen.

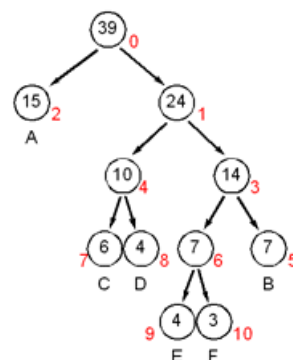


Im Gegensatz zum herkömmlichen (links) wird beim adaptiven Verfahren der Code ständig verändert

Diese Vorgehensweise läßt sich nicht nur bei der Huffman-Kodierung anwenden, wie wir sehen werden, läßt sie sich hier aber besonders günstig einsetzen.

Aktualisieren des Baumes

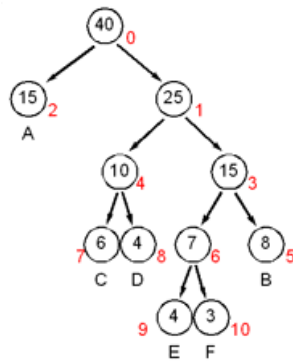
Den Baum nach jedem verarbeiteten Zeichen neu zu konstruieren ist dabei glücklicherweise nicht nötig, würde es doch Kodierung und Dekodierung extrem verlangsamen. Statt dessen nutzt man eine besondere Eigenschaft von zu einem Huffman-Code gehörigen Bäumen: Es existiert eine Numerierung der Knoten, so daß erstens die Numerierung eine der Gewichtung entsprechende Sortierung der Knoten darstellt, und zweitens jeweils zwei Knoten mit gleichem Vorgänger in der Sortierung aufeinander folgen:



Numerierung der Knoten (rot)

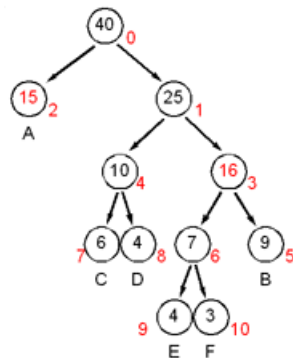
Daß ein Huffman-Baum eine solche Numerierung besitzt, läßt sich leicht dadurch zeigen, daß man die Knoten eben mit der höchsten Nummer beginnend in der Reihenfolge numeriert, in der man sie zusammenfaßt. Wichtig ist aber vor allem, daß, solange man solch eine Numerierung ermöglicht, ein gültiger Baum vorliegt.

Das ermöglicht einem, einen gültigen Baum sehr schnell umzustrukturieren: Wird ein Zeichen kodiert oder dekodiert, werden vom entsprechenden Blatt beginnend die Häufigkeiten bis zur Wurzel schrittweise erhöht.

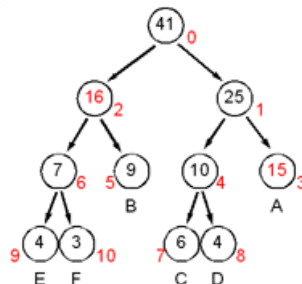


Nach Kodierung eines B sind die Eigenschaften nicht verletzt

Wird dabei die Eigenschaft verletzt, daß die Numerierung eine Sortierung darstellt, müssen zwei Knoten vertauscht werden:



Nach einem weiteren B müssen die Knoten mit den Nummern 2 und 3 vertauscht werden



Nach dem Vertauschen der beiden Knoten ist der Baum wieder gültig

Rescaling

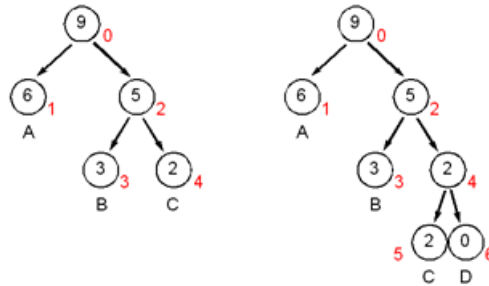
Zunächst aus einer praktischen Not heraus ist man dennoch gezwungen, in regelmäßigen Abständen den Baum neu zu konstruieren: Die Zähler für die Häufigkeiten müssen begrenzt werden, zum einen, weil die gewählten Datentypen immer nur einen begrenzten Wertebereich haben, zum zweiten garantiert eine Begrenzung der Zähler eine Begrenzung der Länge der Codes, die bei der Implementierung der Bitoperationen oft notwendig ist. Deshalb wird nun in regelmäßigen Abständen ein 'Rescaling' der Häufigkeiten vorgenommen, d.h. die Häufigkeiten werden, sobald ihre Summe einen bestimmten Wert erreicht, durch eine Integerdivision halbiert. Dabei muß, weil die Eigenschaften des Huffman-Baumes dann nicht mehr zugesichert werden können, der gesamte Baum neu konstruiert werden. In den üblichen Abständen verlangsamt das die Kodierung aber nur unwesentlich.

Die Reskalierung hat einen interessanten Nebeneffekt: Ein Häufiges Auftreten bestimmter Zeichen an einzelnen Stellen einer Nachricht verblaßt gewissermaßen mit jeder Reskalierung; wenig zurückliegende Vorkommnisse bekommen so ein höheres Gewicht. Auch wenn sich dieser Effekt mathematisch schwer fassen läßt, führt er doch eher zu einer weiteren Verbesserung des Verfahrens,

weil Zeichen in der Praxis oft eine lokal erhöhte Wahrscheinlichkeit aufweisen.

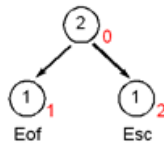
Initialisierung und Einfügen

Ließe sich der Baum nur umordnen, müssten von Anfang an Blätter für alle möglichen Zeichen im Baum vorhanden sein. Damit würde er aber gezwungenermaßen auch Zeichen enthalten, die eventuell nie auftreten und für sie Codes reservieren. Eine wesentlich bessere Lösung ist es dagegen, im Baum ein Zeichen bereitzustellen, das einem bisher nicht vorhandenen Zeichen vorangestellt werden kann (sog. Escape-Sequenz). Tritt ein Zeichen zum ersten Mal beim Kodieren auf, wird es kodiert als Kode des Sonderzeichens gefolgt von dem unkodierten Zeichen. Danach wird das neue Zeichen in den Baum eingefügt, indem der Knoten mit der höchsten Nummer aufgespalten wird:



Das Zeichen D wird aufgenommen (vor dem Aktualisieren des Zählers)

Zu Beginn der Kodierung und Dekodierung muss also kein Zeichen aus der Eingabe im Baum integriert sein, neben dem Escape-Zeichen muss allerdings noch ein anderes Sonderzeichen zur Verfügung stehen, nämlich eines, um das Ende der Nachricht zu kodieren, somit sieht der Baum zu Beginn so aus:



Initialbaum, Eof=End of file, Esc=Escape

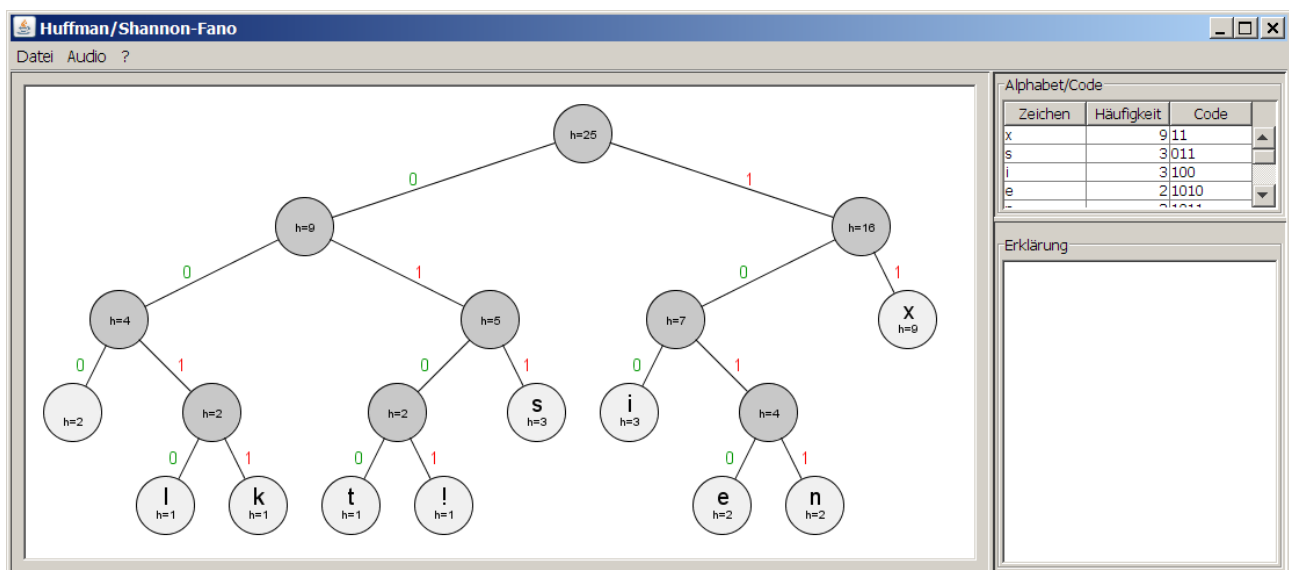
Beispiel zum adaptiven Huffman-Code:

Eingabe-Text: „**sein seil sinkt**“ (natürlich völlig sinnfrei)

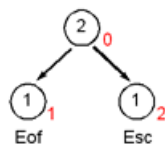
Häufigkeiten (nur zur Kontrolle):

iii	3
sss	3
blank	2
ee	2
nn	2
k	1
l	1
t	1
	15

Erwartetes Ergebnis:



Initialbaum:



Vorgehensweise:

1. Initialbaum zeichnen, Knoten nummerieren, EOF und NYT (=Esc) bekommen Häufigkeit=1 sowie die o.a. Ordnungsnummern.
2. <NYT><neuer Buchstabe> hinzufügen: Den Knoten mit der höchsten Ordnungsnummer splitten, in diesem Fall ist das NYT. Der Knoten behält die Ordnungsnummer, darunter werden Nr. 3 und 4 angehängt, 3 = NYT, 4 = s. NYT Häufigkeit + 1, s Häufigkeit = 1.

3. Bekannter Code: Knoten suchen, Häufigkeit erhöhen.
4. Der Baum muss sortiert bleiben, d.h. je kleiner die Ordnungsnummer, desto größer die Häufigkeit. Dies jeweils von hinten nach vorne überprüfen, und wenn die Sortier-Reihenfolge nicht mehr stimmt, zwei Knoten entsprechend austauschen. Die Knoteninhalte bzw. die unter dem Knoten hängenden Unterknoten / Blätter werden bewegt, die Ordnungsnummern der getauschten Knoten bleiben bestehen!
5. Nach dem Tauschen die Häufigkeiten neu berechnen und ggf. nochmal die Sortierung prüfen und Knoten tauschen.
6. Je Schritt einen neuen Baum zeichnen, bis der komplette Text verarbeitet ist:

NYTsNYTeNYTiNYTnNYT seiNYTl sinNYTkNYTtEOF

7. Code-Tabelle erstellen; zu erwarten ist:

NYT	9	11
s	3	11
i	3	100
e	2	1010
n	2	1011
(leer)	2	0
l	1	10
k	1	11
t	1	100
EOF	1	101

Hinweis: Das Codieren dieses kurzen Textes mit Hilfe handgezeichneter Bäume ist sehr aufwändig, fehleranfällig und eventuell frustrierend, aber wie man so sagt:

Where the going gets rough, the tough get going!